

Práctica 1 Aprendizaje Automático

Antonio Álvarez Caballero

Generación y visualización de datos

Ejercicio 1

En primer lugar, creamos una función que crea un `data.frame` con valores aleatorios según una distribución uniforme. N es el número de filas del `data.frame`, dim el número de columnas y $rango$ el rango donde estarán los valores.

```
simula_unif <- function(N, dim, rango){  
  res <- data.frame(matrix(nrow = N, ncol = dim))  
  
  for(i in 1:N){  
    res[i,] <- runif(dim, rango[1], rango[length(rango)])  
  }  
  
  names(res) <- c("X", "Y")  
  
  return(res)  
}
```

Ejercicio 2

Del mismo modo creamos una función análoga para la distribución *normal* o *gaussiana*.

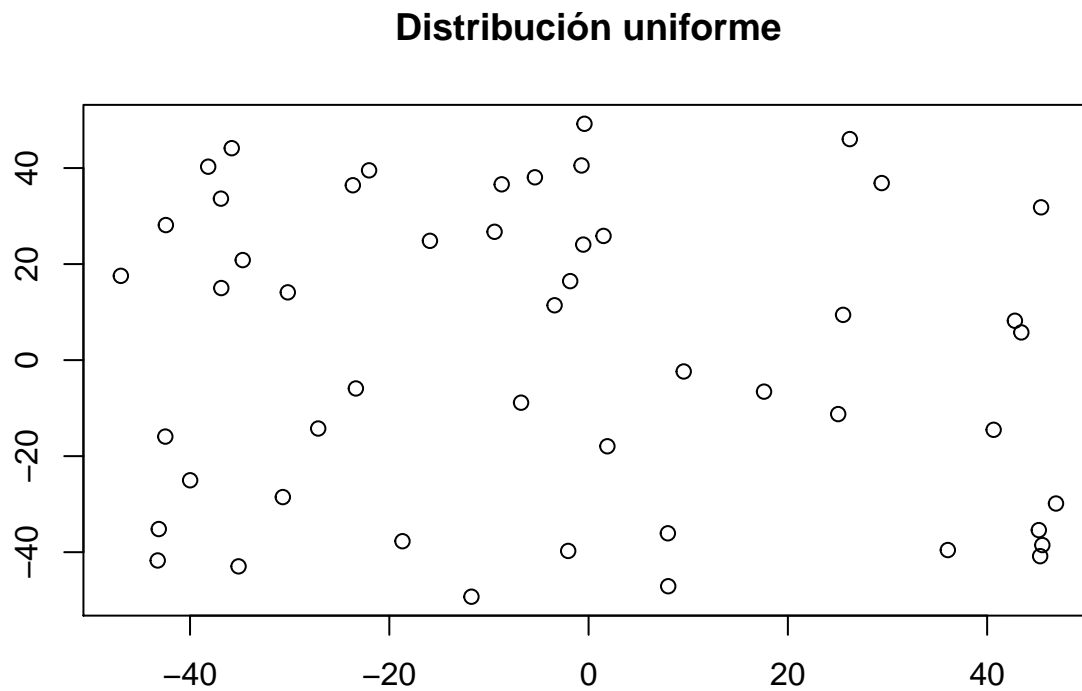
```
simula_gauss <- function(N, dim, sigma){  
  res <- data.frame(matrix(nrow = N, ncol = dim))  
  
  for(i in 1:N){  
    res[i,] <- rnorm(dim, sd = sqrt(sigma))  
  }  
  
  names(res) <- c("X", "Y")  
  
  return(res)  
}
```

Ejercicios 3 y 4

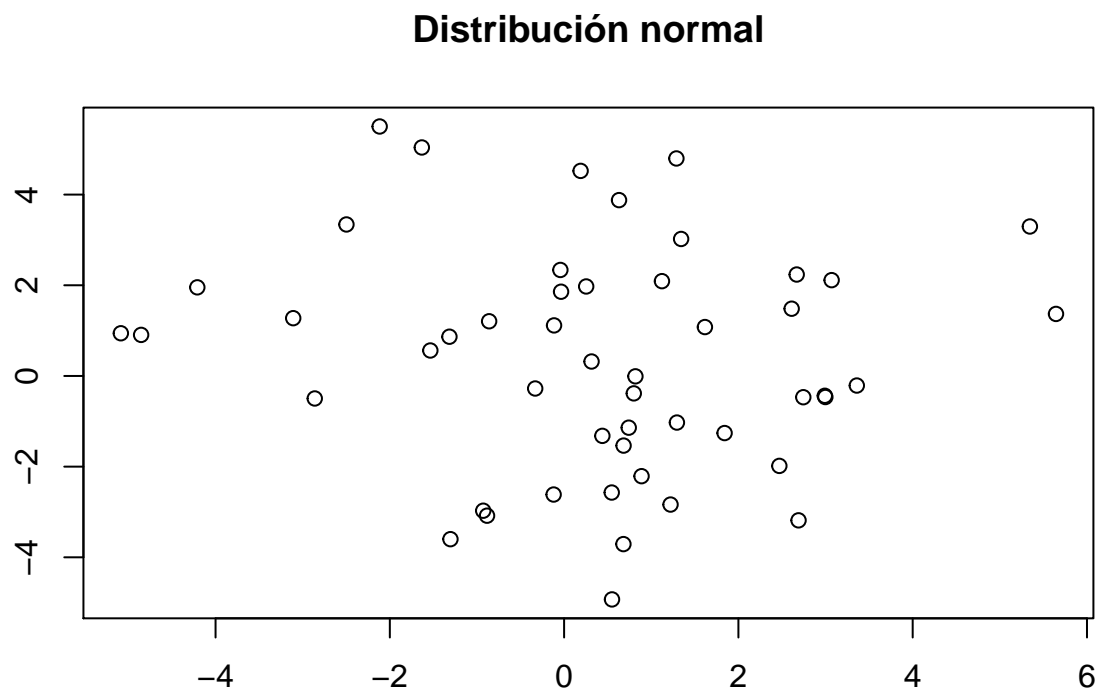
Ahora asignamos los resultados a sendos objetos del tipo *data.frame* y las dibujamos.

```
muestra.uniforme <- simula_unif(50, 2, -50:50)  
muestra.gaussiana <- simula_gauss(50, 2, 5:7)
```

```
plot(muestra.uniforme, main = "Distribución uniforme", xlab = "", ylab = "")
```



```
plot(muestra.gaussiana, main = "Distribución normal", xlab = "", ylab = "")
```



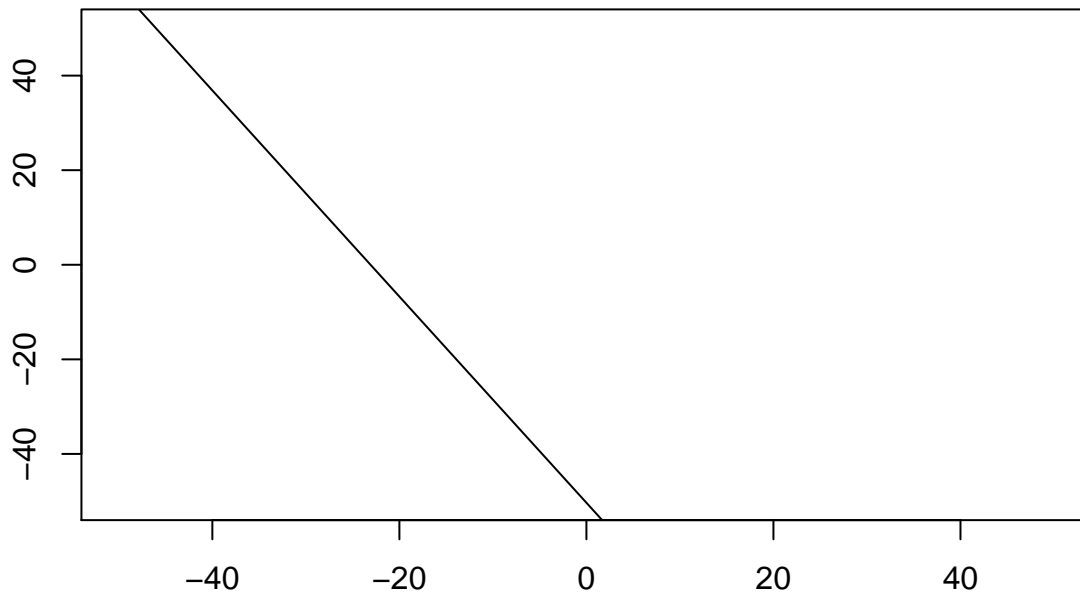
Ejercicio 5

Ahora escribimos una pequeña función para calcular una recta dados dos puntos. Daremos la recta con la ecuación punto pendiente, por lo que tenemos que calcular la pendiente y la desviación.

```
simulaRecta <- function(intervalo){  
  A <- runif(2, intervalo[1], intervalo[length(intervalo)])  
  B <- runif(2, intervalo[1], intervalo[length(intervalo)])  
  
  m <- (A[2] - B[2]) / (A[1] - B[1])  
  b <- A[2] - m * A[1]  
  
  return(c(b,m))  
}
```

Generamos una recta aleatoria en $[-50, 50]$

```
rectaPrueba <- simulaRecta(-50:50)  
  
plot(1, type="n", xlab="", ylab="", xlim=c(-50, 50), ylim=c(-50, 50))  
abline(coef = rectaPrueba)
```



Ejercicio 6

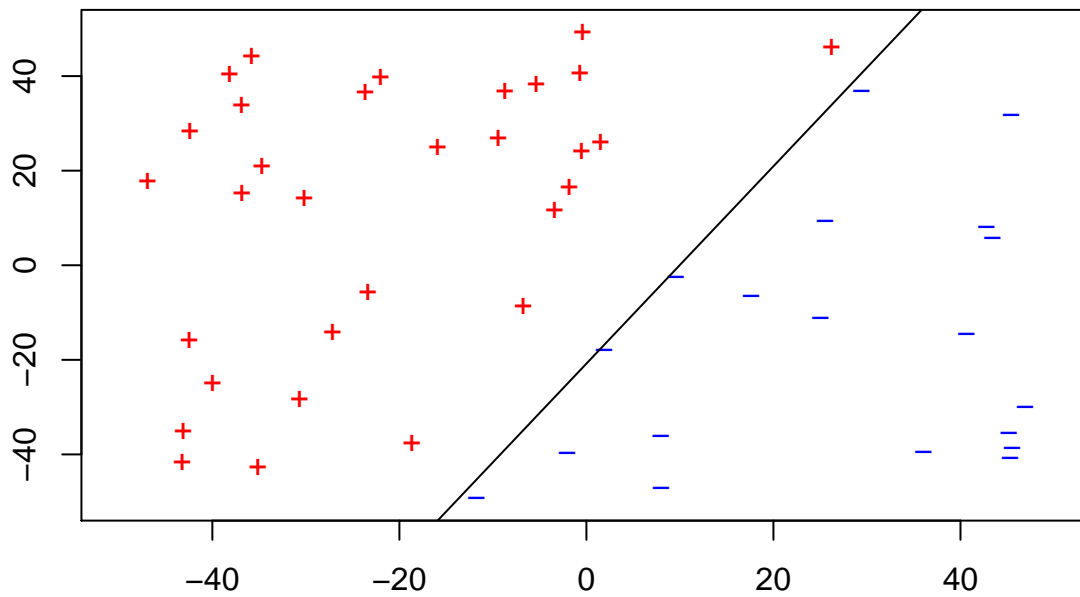
Ahora clasificamos los datos de la muestra uniforme según otra recta aleatoria.

```
rectaClasificacion <- simulaRecta(-50:50)

plot(1, type="n", xlab="", ylab="", xlim=c(-50, 50), ylim=c(-50, 50))

points(subset(muestra.uniforme, Y - rectaClasificacion[2]*X - rectaClasificacion[1] > 0 ),
       pch = "+", col = "red")
points(subset(muestra.uniforme, Y - rectaClasificacion[2]*X - rectaClasificacion[1] <= 0 ),
       pch = "-", col = "blue")

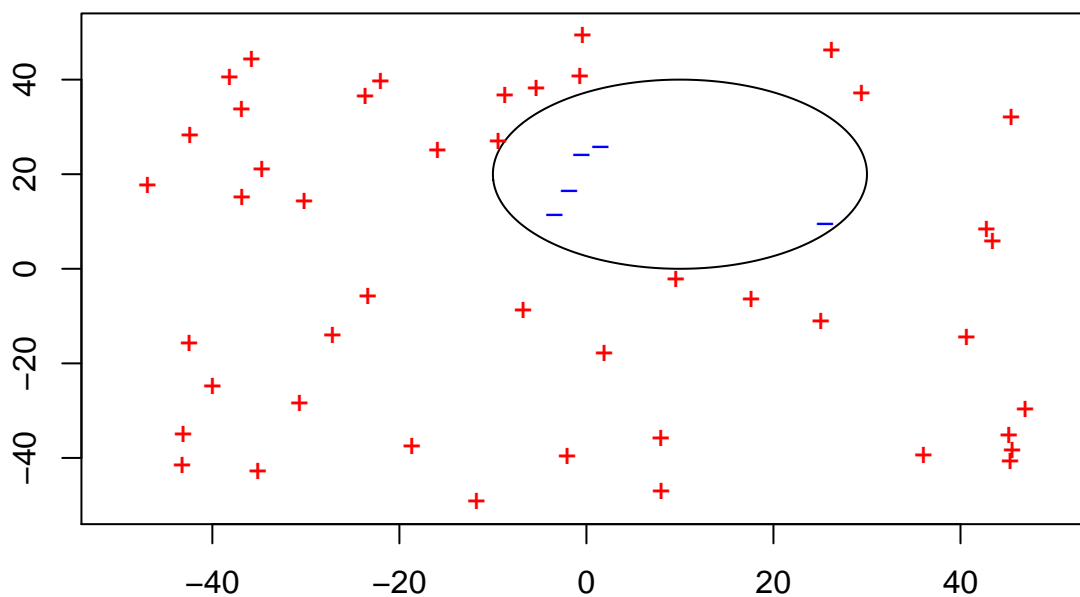
abline(coef = rectaClasificacion)
```



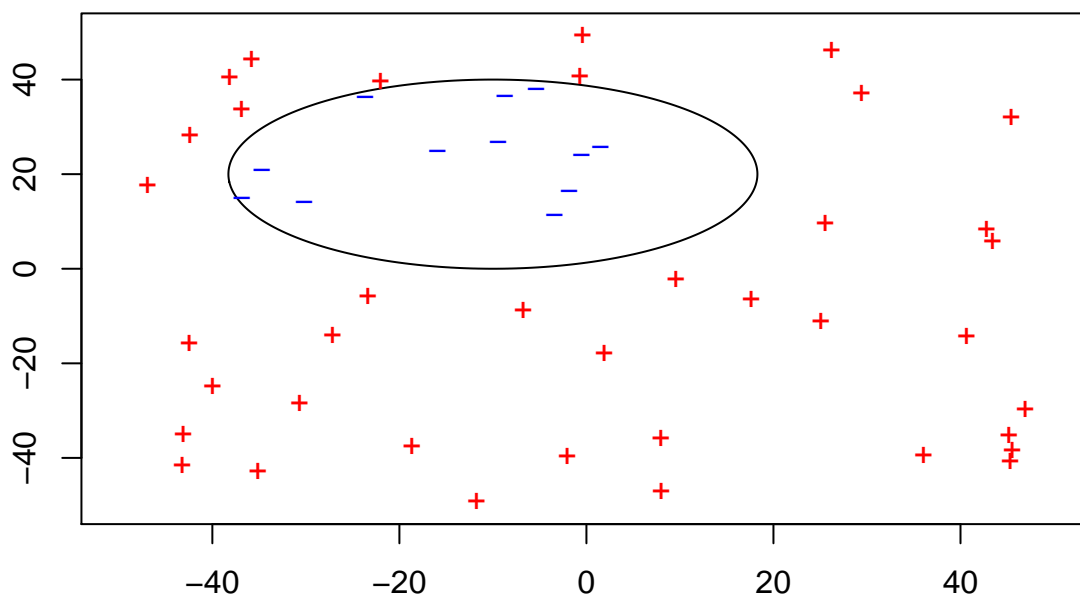
Ejercicio 7

Una vez hemos clasificado los datos con una recta, clasificamos con otro tipo de funciones.

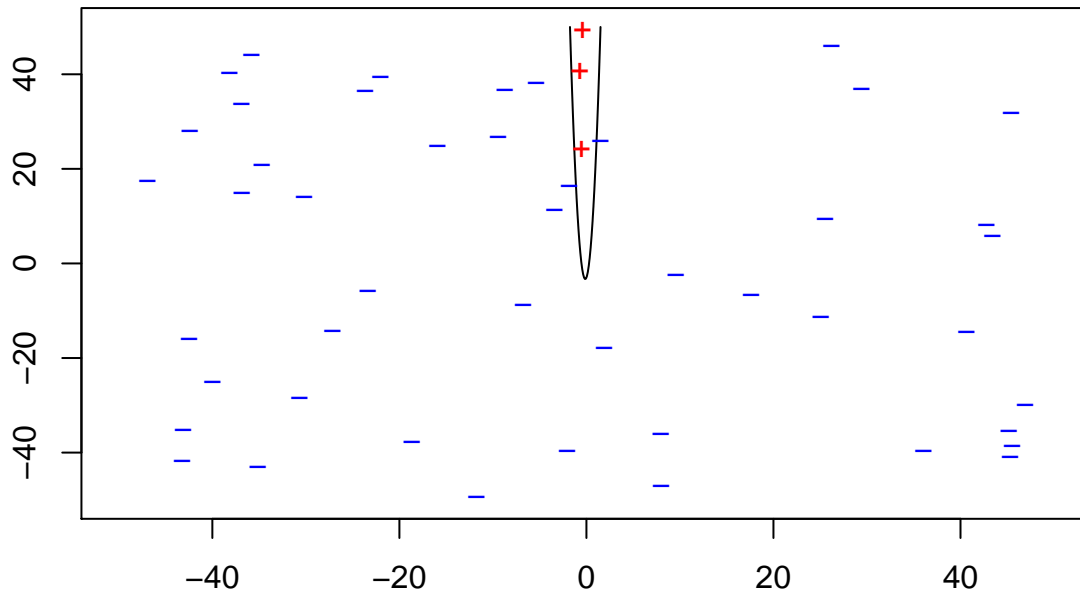
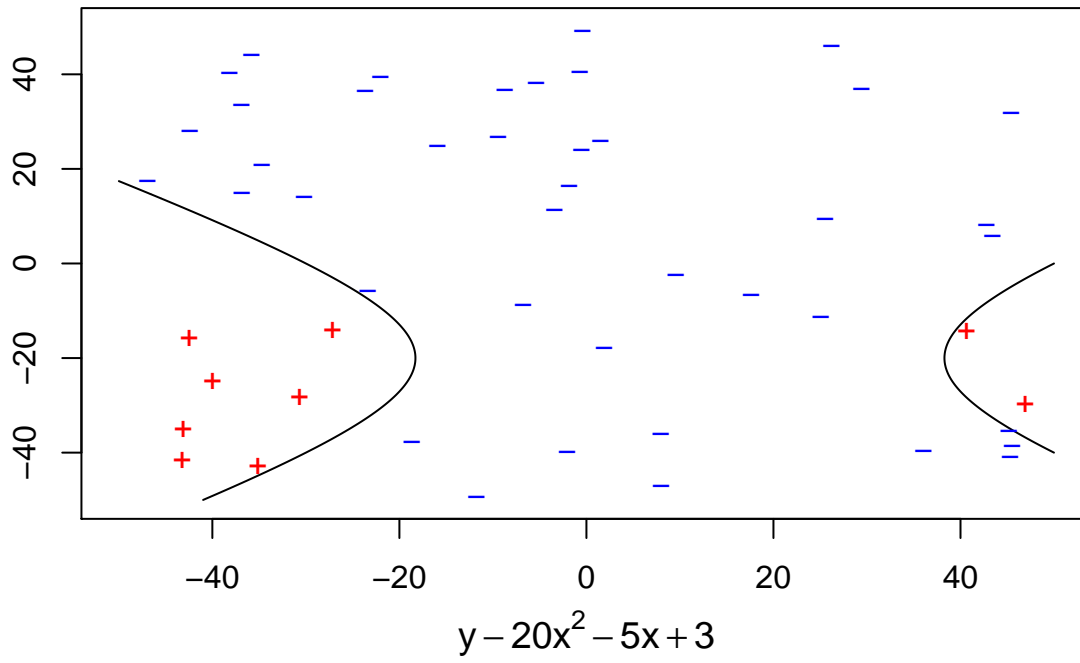
$$(x-10)^2 + (y-20)^2 - 400$$



$$0.5(x+10)^2 + (y-20)^2 - 400$$



$$0.5(x-10)^2 - (y+20)^2 - 400$$



Añadir aquí conclusiones de las regiones nuevas comparadas con la lineal.

Ejercicio 8

Holi